UCS 1617 – MINI PROJECT
STUDENTS INFORMATION SYSTEM

STATE CHART & ACTIVITY DIAGRAM

TEAM MEMBERS :

- SHRINISHA N (18 5001 148)
- SRIPRABHA AR (18 5001 167)
- SUBA SHREE V S (18 5001 171)

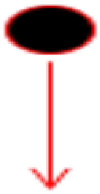
Aim :

To construct a state machine and activity diagram for Student Information system.

UML Notations for State Machine Diagram :

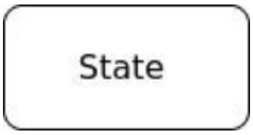
1. Initial State:

We use a black filled circle to represent the initial state of a System or a class.



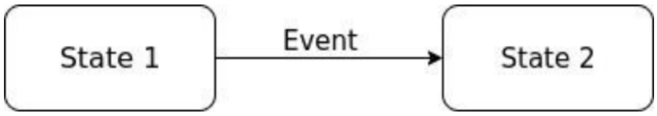
2. State:

We use a rounded rectangle to represent a state. A state represents the conditions or circumstances of an object of a class at an instant of time.



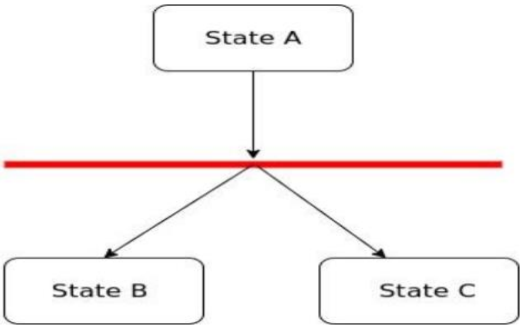
3. Transition:

We use a solid arrow to represent the transition or change of control from one state to another. The arrow is labelled with the event which causes the change in state.



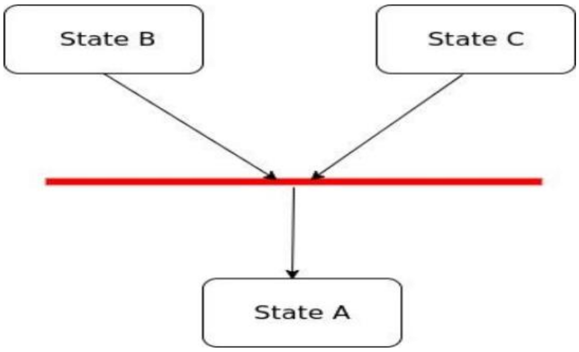
4. Fork:

We use a rounded solid rectangular bar to represent a Fork notation with incoming arrow from the parent state and outgoing arrows towards the newly created states. We use the fork notation to represent a state splitting into two or more concurrent states.



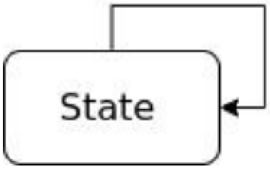
5. Join:

We use a rounded solid rectangular bar to represent a Join notation with incoming arrows from the joining states and outgoing arrow towards the common goal state. We use the join notation when two or more states concurrently converge into one on the occurrence of an event or events.

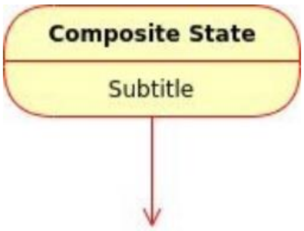


6. Self Transition:

We use a solid arrow pointing back to the state itself to represent a self transition. There might be scenarios when the state of the object does not change upon the occurrence of an event. We use self transitions to represent such cases.



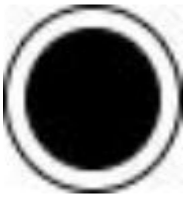
7. Composite State:



We use a rounded rectangle to represent a composite state also.We represent a state with internal activities using a composite state.

8. Final State:

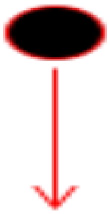
We use a filled circle within a circle notation to represent the final state in a state machine diagram.



UML Notations for Activity Diagram:

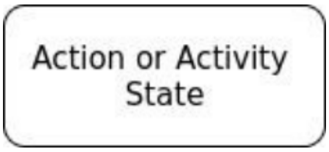
1. Initial State:

The starting stage before an activity takes place is depicted as the initial state.A process can have only one initial state unless we are depicting nested activities. We use a black filled circle to depict the initial state of a system.



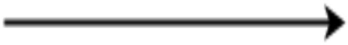
2. Action or Activity state:

An activity represents execution of an action on objects or by objects. We represent an activity using a rectangle with rounded corners. Basically any action or event that takes place is represented using an activity.



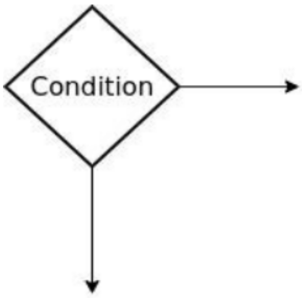
3. Action Flow or Control Flow:

Action flows or Control flows are also referred to as paths and edges. They are used to show the transition from one activity state to another.



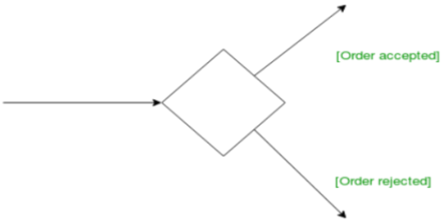
4. Decision node and branching:

When we need to make a decision before deciding the flow of control, we use the decision node.The outgoing arrows from the decision node can be labelled with conditions or guard expressions.It always includes two or more output arrows.



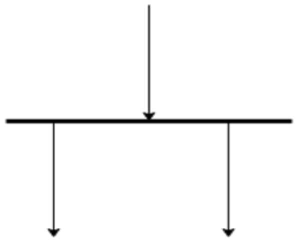
5. Guards:

A Guard refers to a statement written next to a decision node on an arrow sometimes within square brackets.The statement must be true for the control to shift along a particular direction. Guards help us know the constraints and conditions which determine the flow of a process.



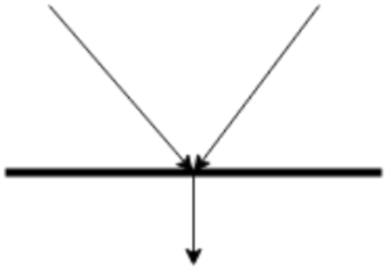
6. Fork:

Fork nodes are used to support concurrent activities. When we use a fork node both the activities get executed concurrently i.e. no decision is made before splitting the activity into two parts. Both parts need to be executed
in case of a fork statement.



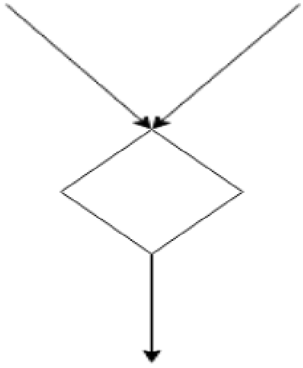
7. Join:

Join nodes are used to support concurrent activities converging into one. For join notations we have two or more incoming edges and one outgoing edge.



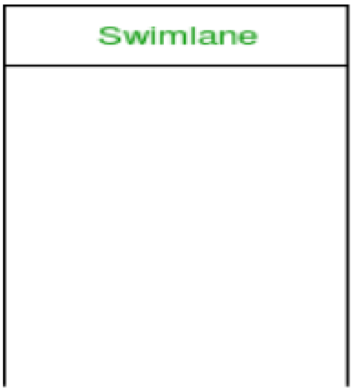
8. Merge or Merge event:

Scenarios arise when activities which are not being executed concurrently have to be merged. We use the merge notation for such scenarios. We can merge two or more activities into one if the control proceeds onto the next activity irrespective of the path chosen.



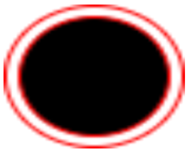
9. Swimlanes:

Swimlanes group related activities into one column or one row. Swimlanes can be vertical and horizontal. Swimlanes are used to add modularity to the activitydiagram.

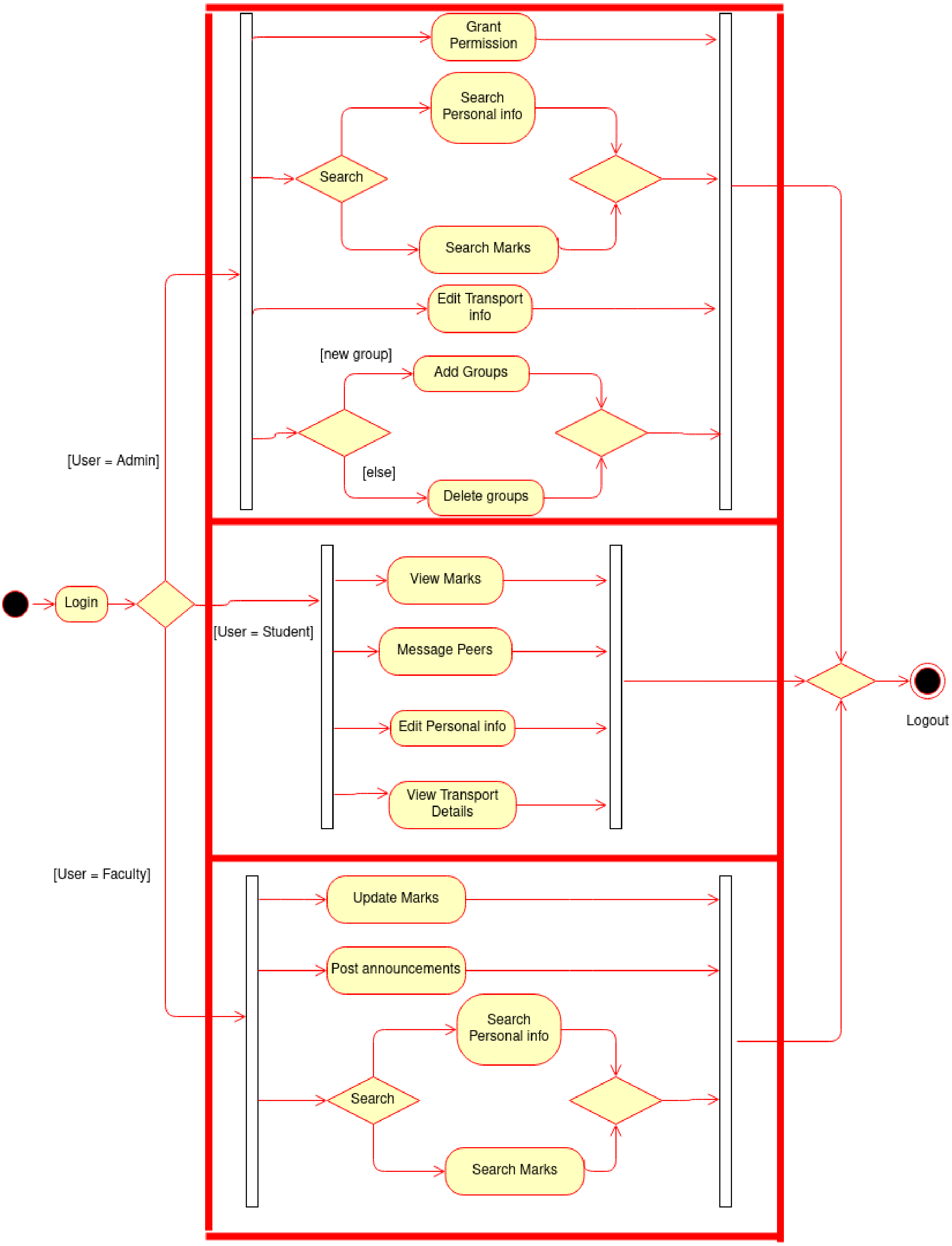


10. Final state or End state:

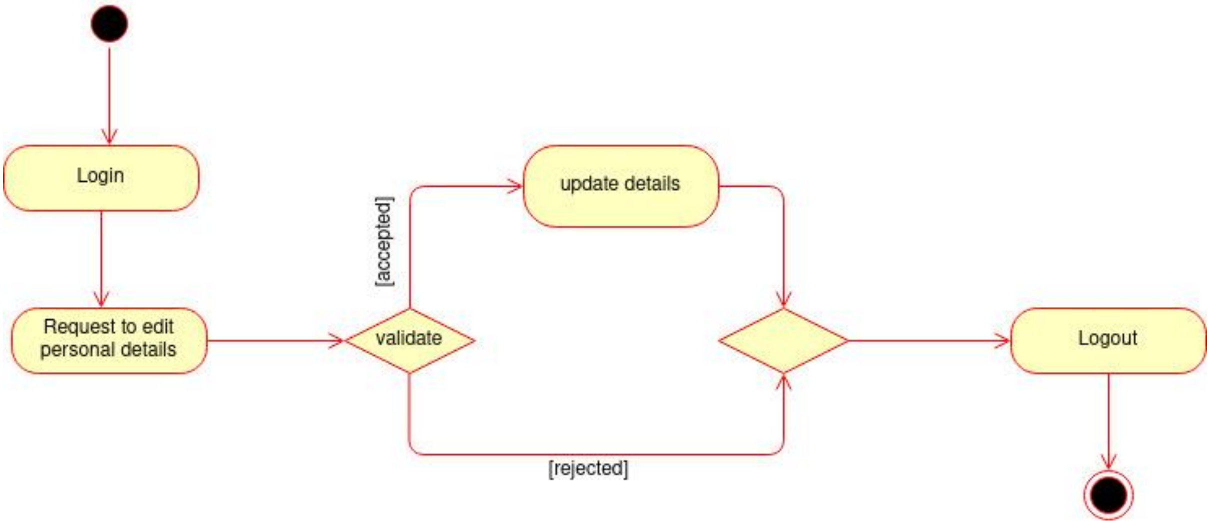
The state which the system reaches when a particular process or activity ends is known as a Final State or End State. We use a filled circle within a circle notation to represent the final state in a state machine diagram. A system or a process can have multiple final states.

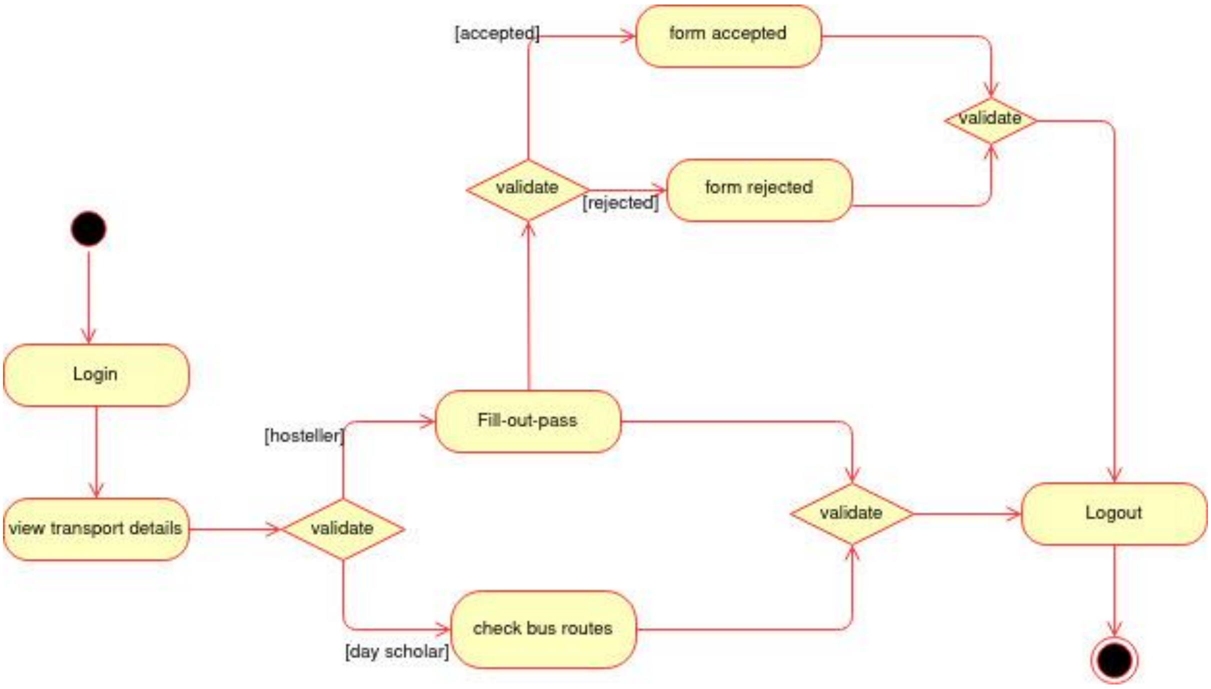


Main Activity Diagram:

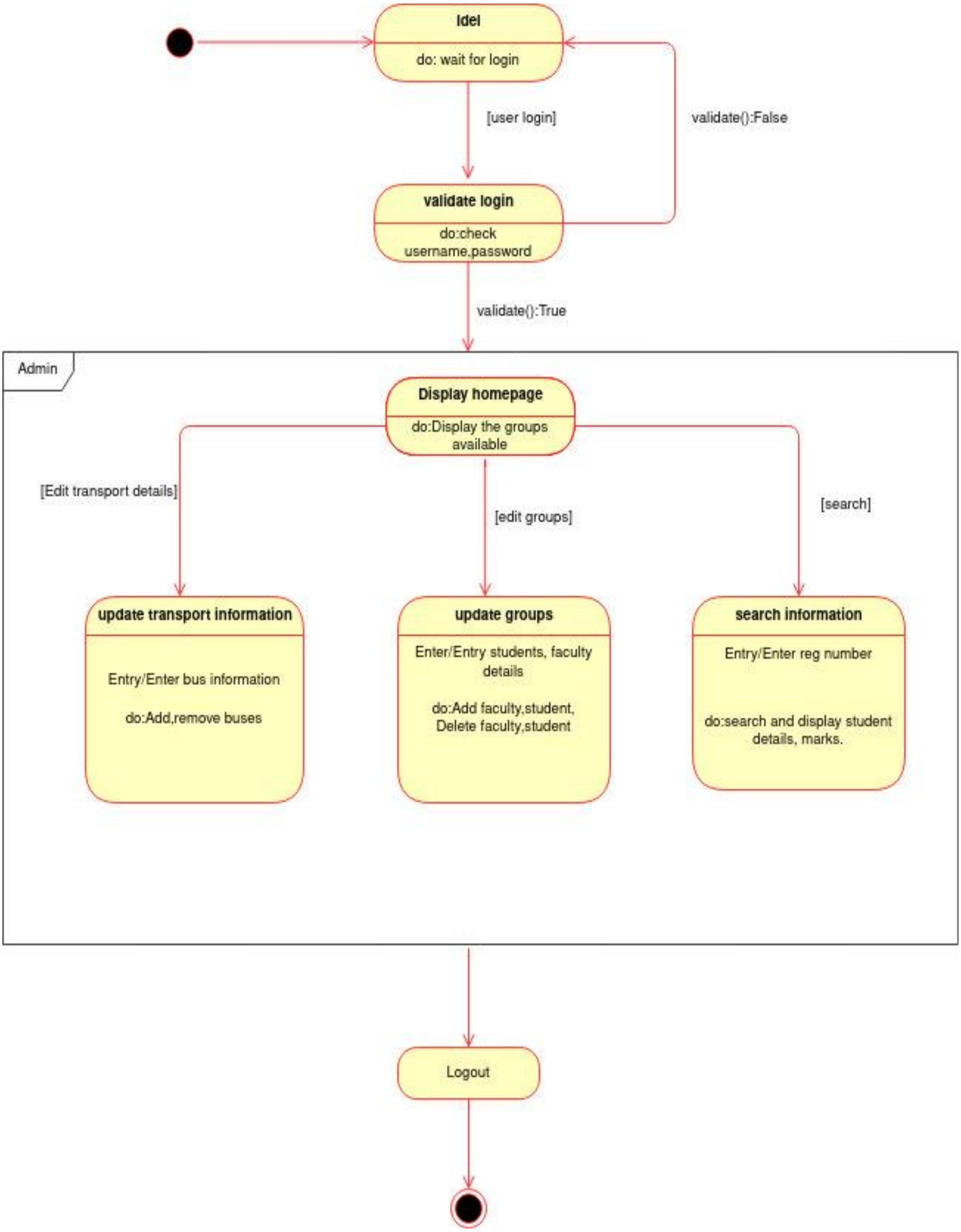


SUB ACTIVITY DIAGRAMS:

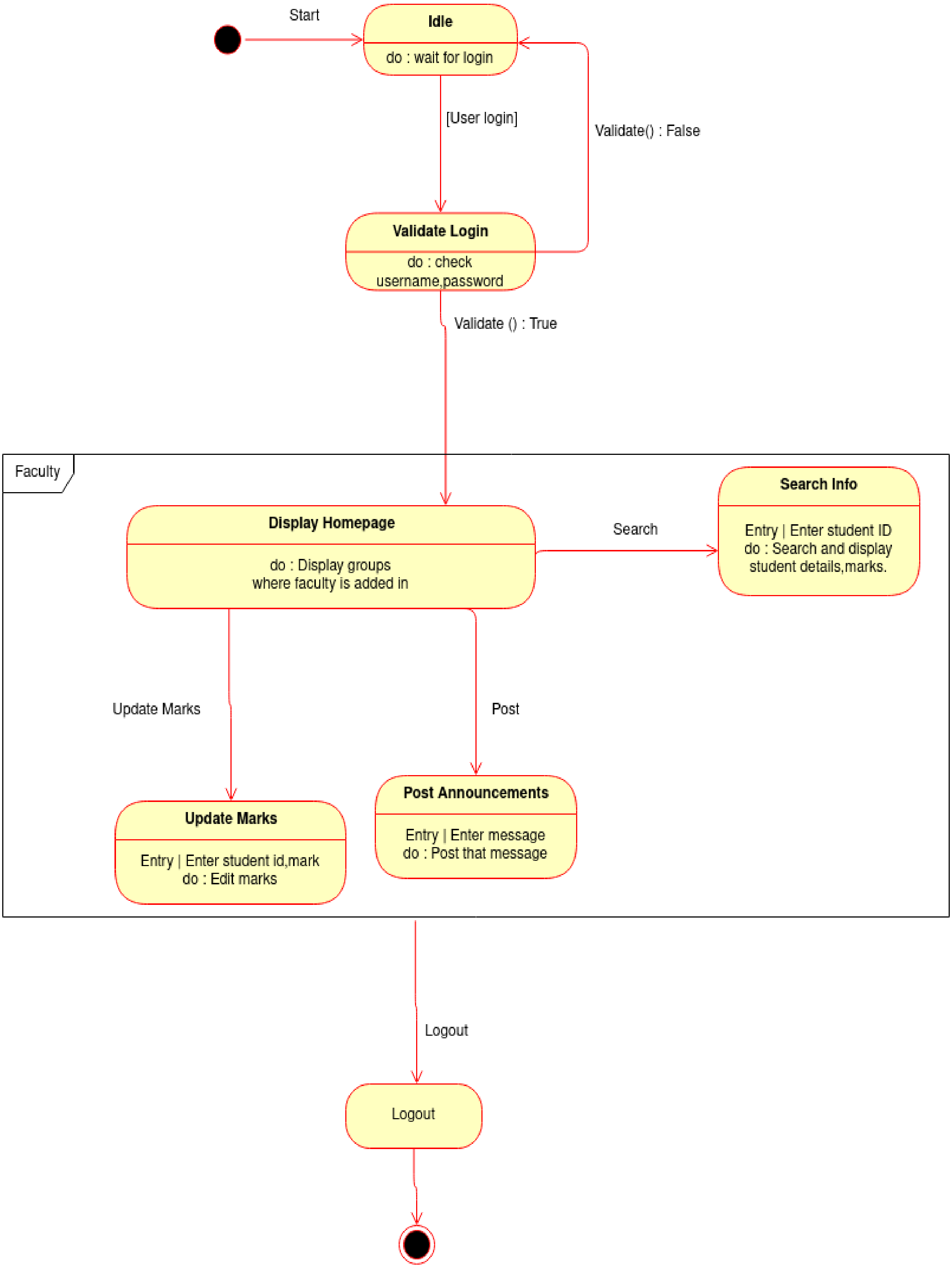

Login
Request to edit personal details
validate
[accepted]
update details
[rejected]
Logout


[accepted]
form accepted
validate
validate
[rejected]
form rejected
Login
[hosteller]
Fill-out-pass
view transport details
validate
validate
Logout
check bus routes
[day scholar]

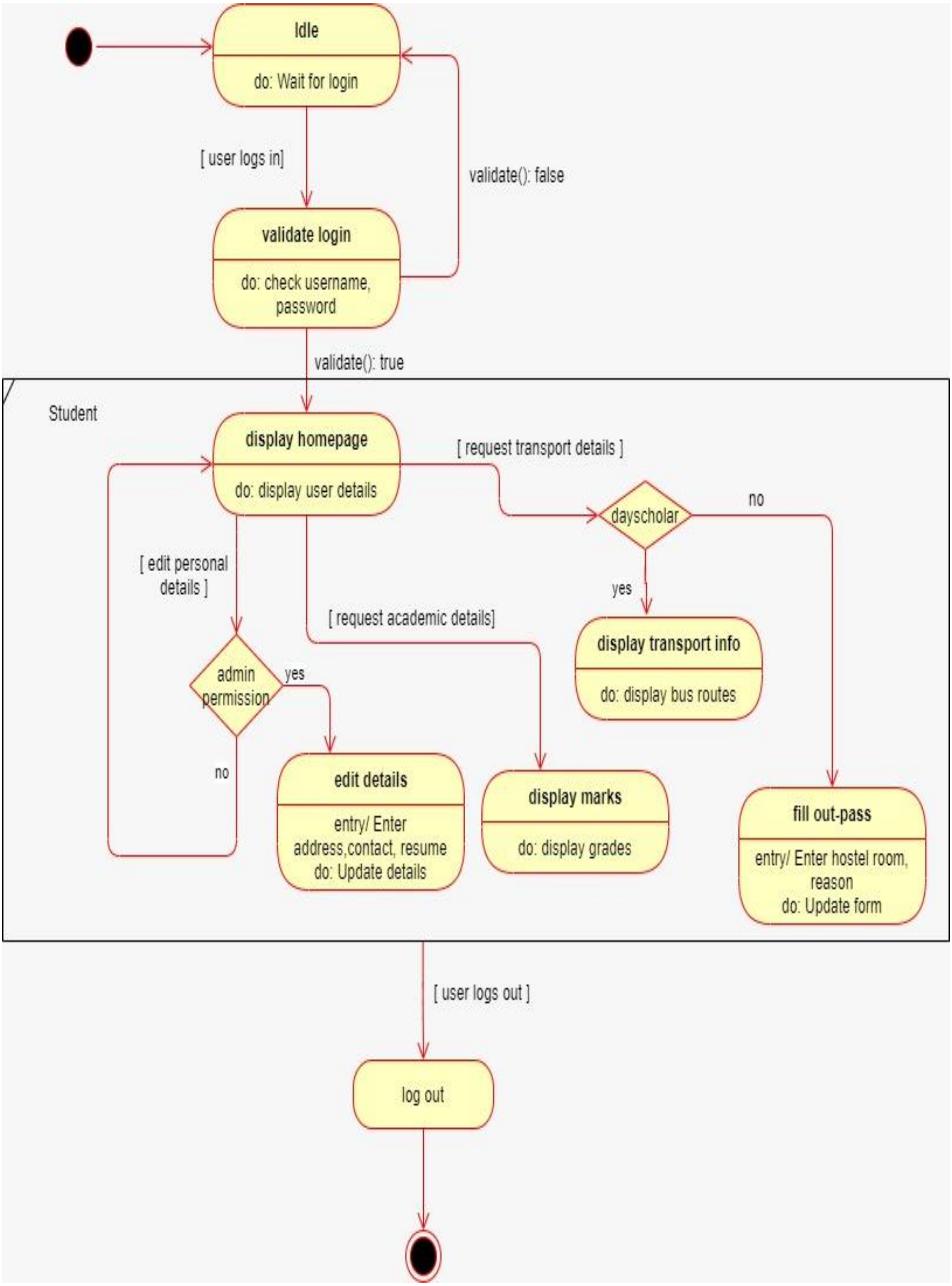
Admin:


Idel
do: wait for login
[user login]
validate():False
validate login
do:check username,password
validate():True
Admin
Display homepage
do:Display the groups available
[Edit transport details]
[edit groups]
[search]
update transport information
Entry/Enter bus information
do:Add,remove buses
update groups
Enter/Entry students, faculty details
do:Add faculty,student, Delete faculty,student
search information
Entry/Enter reg number
do:search and display student details, marks.
Logout

Faculty:


Start
Idle
do : wait for login
[User login]
Validate() : False
Validate Login
do : check username,password
Validate () : True
Faculty
Display Homepage
do : Display groups where faculty is added in
Search
Search Info
Entry | Enter student ID
do : Search and display student details,marks.
Update Marks
Post
Post Announcements
Entry | Enter message
do : Post that message
Update Marks
Entry | Enter student id,mark
do : Edit marks
Logout
Logout

Student:



Idle
do: Wait for login
[user logs in]
validate(): false
validate login
do: check username, password
validate(): true
Student
display homepage
do: display user details
[request transport details]
dayscholar
no
yes
display transport info
do: display bus routes
[edit personal details]
[request academic details]
admin permission
yes
no
edit details
entry/ Enter address,contact, resume
do: Update details
display marks
do: display grades
fill out-pass
entry/ Enter hostel room, reason
do: Update form
[user logs out]
log out